

CPSC 2221 – QUIZ 2

Date: Nov 16, 2022

Time: 1 Hours 45 Minutes Due: 10.25 AM

Marks: 30 Marks

- **The Quiz 2 consists of 2 sections.**
- **One is creating the Database with 3 tables and the another is answering queries based on the created table. Submit the file after finishing the question on Moodle link for Midterm 2.**
- **The exam is open book and slides. Laptop with PostgreSQL is allowed.**
- **No copying code from the internet is allowed. Postgres documentation can be used.**
- **Separate spaces provided for query and screenshots of the answer.**
- **Not all questions need screenshots. Only answer what is asked in the question. If you do not find the box to paste the screenshot, then it is not needed.**

Consider a Superstore Database which consists of 3 tables, Orders, Returns and Managers. The CSV files have been provided along with this DOC file in the Midterm 2 Link in the Moodle. Answer the questions as below:

1. Create the three tables with their SQL queries in a newly created Database Superstore and paste the code in text format below. Make sure to assign primary and foreign keys as needed. Be careful while using the Unique and NOT NULL constraints. Use them only when required. Use correct datatype for each attribute in an Entity. Use PG-ADMIN to create these tables, as the tables will be needed to answer questions in the question 2.

Write Create Orders Table Query: **[6 Marks]**

[Hint: Order ID is not a Foreign key. Be careful while defining OrderID from Orders and Returns table]

```
CREATE TABLE Superstore_Orders(  
    RowID INT UNIQUE NOT NULL,  
    OrderPriority VARCHAR(255),  
    Discount NUMERIC,  
    UnitPrice NUMERIC,  
    ShippingCost NUMERIC,  
    CustomerID INT,  
    CustomerName Varchar(255),  
    ShipMode Varchar(255),  
    CustomerSegment Varchar(255),  
    ProductCategory Varchar(255),  
    ProductSub_Category Varchar(255),  
    ProductContainer Varchar (255),  
    ProductBaseMargin NUMERIC,  
    Region Varchar(35),  
    Province Varchar(45),  
    City Varchar(35),  
    PostalCode INT,  
    OrderDate Date,  
    ShipDate Date,  
    Profit NUMERIC,  
    Quantity INT,  
    Sales NUMERIC,  
    OrderId INT,  
    FOREIGN KEY(Region) REFERENCES  
    Superstore_Managers(Region),  
    PRIMARY KEY(RowID))
```

Write Create Returns Table Query: **[2 Marks]**

```
CREATE TABLE Superstore_Returns(  
    Order_Id INT,  
    Status Varchar(35),  
    PRIMARY KEY(Order_Id)  
);
```

Write Create Managers Table Query: **[2 Marks]**

```
CREATE TABLE Superstore_Managers(  
    Region Varchar(35) NOT NULL,  
    Manager Varchar(15),  
    PRIMARY KEY (Region)  
);
```

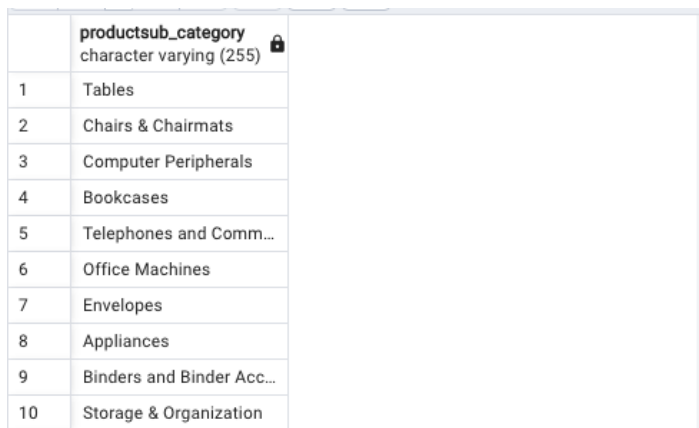
2. Use the created table as in Question 1, solve the problems as mentioned below. You will have to import the respective CSV files of the above created tables as without them, it is impossible to solve the questions below. If you are not able to upload the files successfully, do not leave the query questions. Just write the query to the best of your knowledge. Do not copy. To be graded for the screenshot answer, you must upload the CSV properly and paste the resulting screenshot of the queries as asked.


- (a) Write Query to Find out which Product Sub-Category has a sum of Shipping Cost to sum of Sales ratio > 0.03.

Write the Query in box below. **[4 Marks]**

```
SELECT productsub_category
FROM superstore_orders
WHERE (shippingcost / sales) > 0.03
GROUP BY productsub_category
```

Paste the screenshot of a portion of the answer below. **[1 Marks]**

A screenshot of a table with 10 rows and 2 columns. The first column contains numbers 1 through 10. The second column contains product sub-categories. The table is titled 'productsub_category' with a subtitle 'character varying (255)' and a lock icon.

	productsub_category character varying (255) 
1	Tables
2	Chairs & Chairmats
3	Computer Peripherals
4	Bookcases
5	Telephones and Comm...
6	Office Machines
7	Envelopes
8	Appliances
9	Binders and Binder Acc...
10	Storage & Organization

- (b) Write Queries to add 2 columns 'Return Status' & 'Days to Ship' to the Orders table.

Write the Add column Queries in box below. **[3 Marks]**

```
ALTER TABLE superstore_orders  
ADD COLUMN "Return Status" VARCHAR(35),  
ADD COLUMN "Days to Ship" INT;
```

	y	sales numeric	orderid integer	Return Status character varying (35)	Days to Ship integer
1	2	5.9	88525	[null]	[null]
2	4	13.01	88522	[null]	[null]
3	7	49.92	88523	[null]	[null]
4	7	41.64	88523	[null]	[null]
5	8	1446.67	88523	[null]	[null]
6	37	2011.67	88524	[null]	[null]
7	12	1451.37	88526	[null]	[null]
8	12	6362.85	90193	[null]	[null]
9	18	113.25	90197	[null]	[null]
10	16	1515.17	90194	[null]	[null]

'Return Status' data comes from the Returns table, after adding the table use Update/CASE/WHEN/THEN to append a Boolean value True or False based on return status from the Returns table. Write Query below. **[3 Marks]**

'Days to Ship' is the difference between Order date and ship date column in the Orders table. Again, use Update to add data to the 'Days to Ship' Column **[2 Marks]**

```
UPDATE Superstore_Orders
SET "Days to Ship" =
CASE
WHEN Orderdate < Shipdate THEN Shipdate - Orderdate
ELSE Orderdate - Shipdate
END;
```

quantity integer	sales numeric	orderid integer	Return Status character varying (35)	Da in
5	1274.79	87373	[null]	
14	93.97	87307	[null]	
5	89.17	89429	[null]	
17	168.44	86720	[null]	
15	1433.08	91519	[null]	
12	129.1	88463	[null]	
5	26.59	88106	[null]	
7	21.53	91332	[null]	

- (c) Create a Function which takes in OrderID column of Orders table as the input and returns back a table with columns return status of the order (True or False), name of the manager handling that Order and order priority of that order.

Write the Query in box below. **[5 Marks]**

```
CREATE OR REPLACE FUNCTION get_Info(orderID INT)
RETURNS TABLE (
  orderStatus bool,
  managerName Varchar(25),
  orderPriority Varchar(25)
)
AS $$
BEGIN
RETURN QUERY
SELECT r.status, m.manager, o.orderpriority
FROM (superstore_orders o INNER JOIN superstore_manager m ON
o.Region = m.Region)
LEFT JOIN superstore_returns r ON o.orderid = r.order_id
END;$$
```

Paste the screenshot for a sample OrderID input to the function. **[2 Marks]**